PelicanHPC Tutorial

July 2008
Michael Creel
Universitat Autònoma de Barcelona

You can check for more recent versions of this document at http://pareto.uab.es/mcreel/PelicanHPC/Tutorial/PelicanTutorial.html

Contents

1. Introduction
2. Initial setup
3. Example software
4. Saving your work
5. Using the make_pelican script

Introduction

PelicanHPC is a rapid (around 5 minutes, when you know what you're doing) means of setting up a high performance computing (HPC) cluster for parallel computing using MPI. This tutorial gives a basic description of what PelicanHPC does, addresses how to use the released CD images to set up a HPC cluster, and gives some basic examples of usage.

Description of PelicanHPC

PelicanHPC is a distribution of GNU/Linux that runs as a "live CD" (or as a virtualization appliance). If the ISO image file is burnt to a CD, the resulting CD can be used to boot a computer. The computer on which PelicanHPC is booted is referred to as the "frontend node", which is the computer that the user interacts with. Once PelicanHPC is running, a script - "pelican_setup" - may be run. This script configures the frontend node as a netboot server. After this has been done, other computers can boot copies of PelicanHPC over the network. These other computers are referred to as "compute nodes". PelicanHPC configures the cluster made up of the frontend node and the compute nodes so that MPI-based parallel computing may be done.

A "live CD" such as PelicanHPC does not use the hard disk of any of the nodes, so it will not destroy or alter your installed operating system. When the PelicanHPC cluster is shut down, all of the computers are in their original state, and will boot back into whatever operating system is installed.

Features

- The frontend node can be a real computer booted using a CD, or a virtual machine that is booted using the CD image file. With this second option, PelicanHPC can be used at the same time as the normal work environment, which may be any of the common operating systems.

- The compute nodes are normally real computers, but they can also be virtual.
- Supports MPI-based parallel computing using Fortran (77, 90), C, C++, and GNU Octave (using MPITB).
- Offers the Open MPI and LAM/MPI implementations of MPI.
- Cluster can be resized to add or remove nodes using the "pelican_restarthpc" command.
- Easily extensible to add packages. Also easily modifiable, since the PelicanHPC CD image is created using a single script that uses the Debian Live system for creating a live CD image. For this reason, the distributed version is basic and lightweight.
- Versions exist for 32 bit CPUs (Pentium 4, Core, Sempron) and for 64 bit CPUs (Opteron, Turion, Core 2, etc.)
- Contains example software: Linpack HPL benchmark and extensive examples that use MPITB for GNU Octave.

Limitations and requirements

- The compute nodes must be booted over the network. This is an option offered by all modern networking devices supplied with motherboards, but it often must be enabled in the BIOS setup. Enable it, and give it higher priority that booting from hard disk or other sources. If you have a network card that won't do netboot, it is possible to work around this using rom-o-matic.
- A PelicanHPC cluster is "volatile". It disappears when the machines are turned off. It is possible (see below) to use hard disk space or other storage to retain work, but this requires some knowledge of the GNU/Linux operating system. If using virtualization, is is possible to take snapshots of the nodes, which allows work to be recovered quickly. PelicanHPC is not meant to be installed as the primary operating system of a computer.
- A PelicanHPC cluster is designed to be used by a single person.

Licensing and Disclaimer

PelicanHPC is a CD image made by running a script (see below). The script is licensed GPL v3. The resulting CD image contains software from the Debian distribution of GNU/Linux, which is subject to the licenses chosen by the authors of that software.

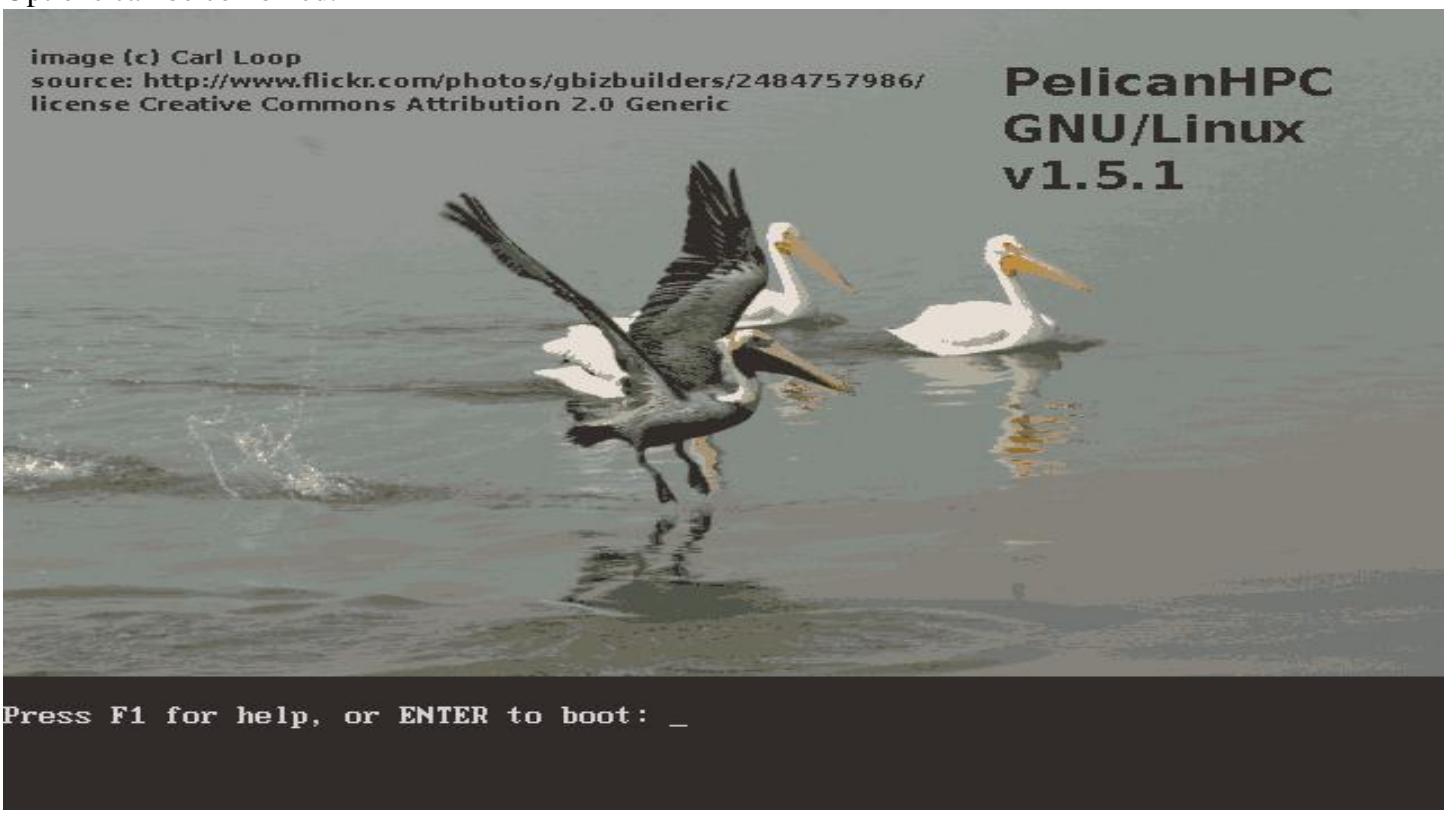
This released PelicanHPC CD images are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Initial setup

The frontend and all compute nodes must be networked together. IMPORTANT: the frontend node will act as a DHCP server, so be sure to isolate the network used for the cluster from other networks, to avoid conflicts with other DHCP servers. If you start handing out IP addresses to your co-workers' computers, they may become annoyed. If the frontend node has multiple network interfaces, you can use one to connect to the cluster and another to connect to the Internet.

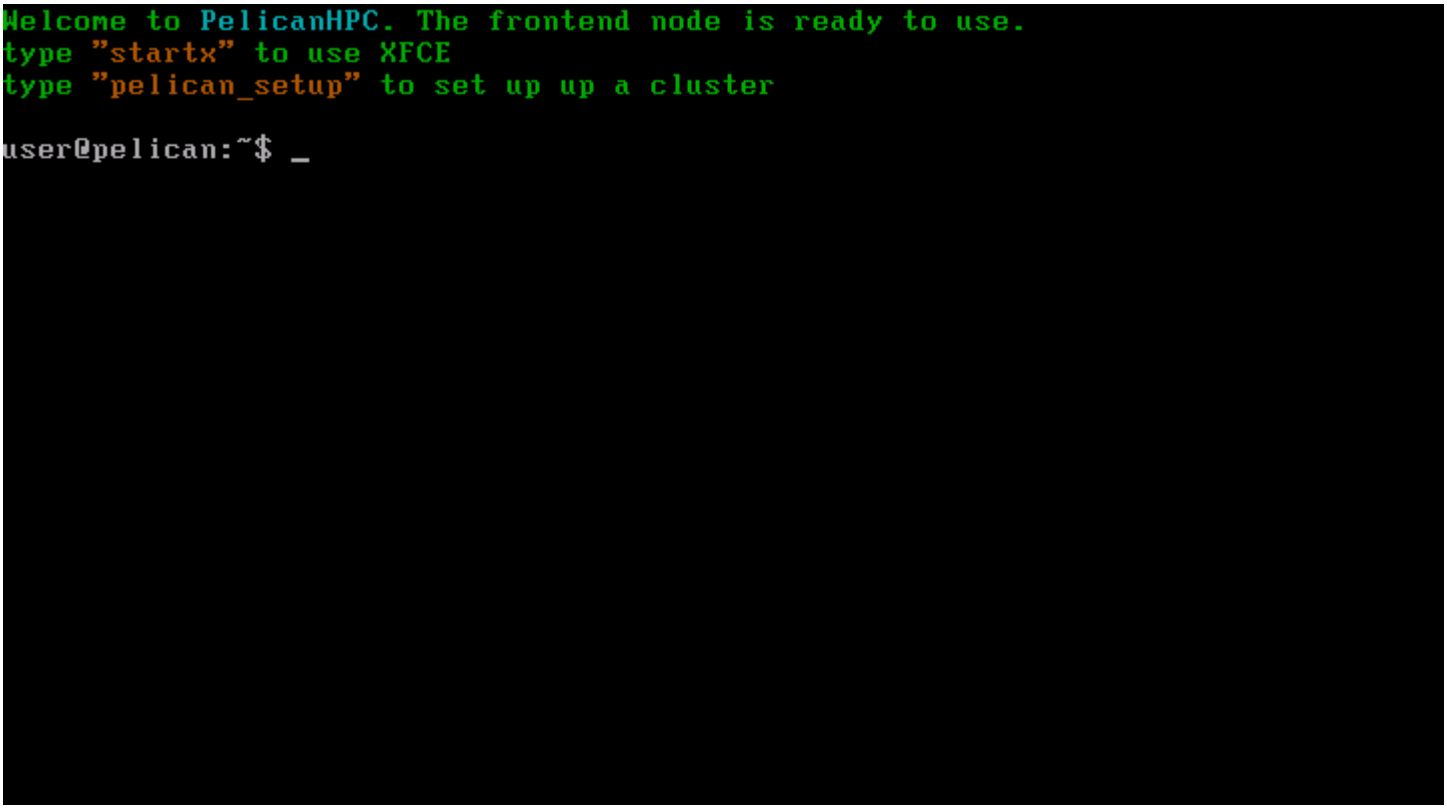
Put the CD in the computer that will be the frontend, and turn it on. Make sure the BIOS setup lets you boot from CD. When you boot up, you'll see something like the following. Either explore the options, or press <Enter> to boot up. For example, I can get a Spanish keyboard by typing "live keyb=es" and <Enter>.

Options can be combined.



There are some examples for GNU Octave using MPITB, and the Linpack HPL benchmark is also available. If you're interested in these, choose yes. If you just want a cluster and you will supply the applications, choose no.

OK, the frontend is all set up:

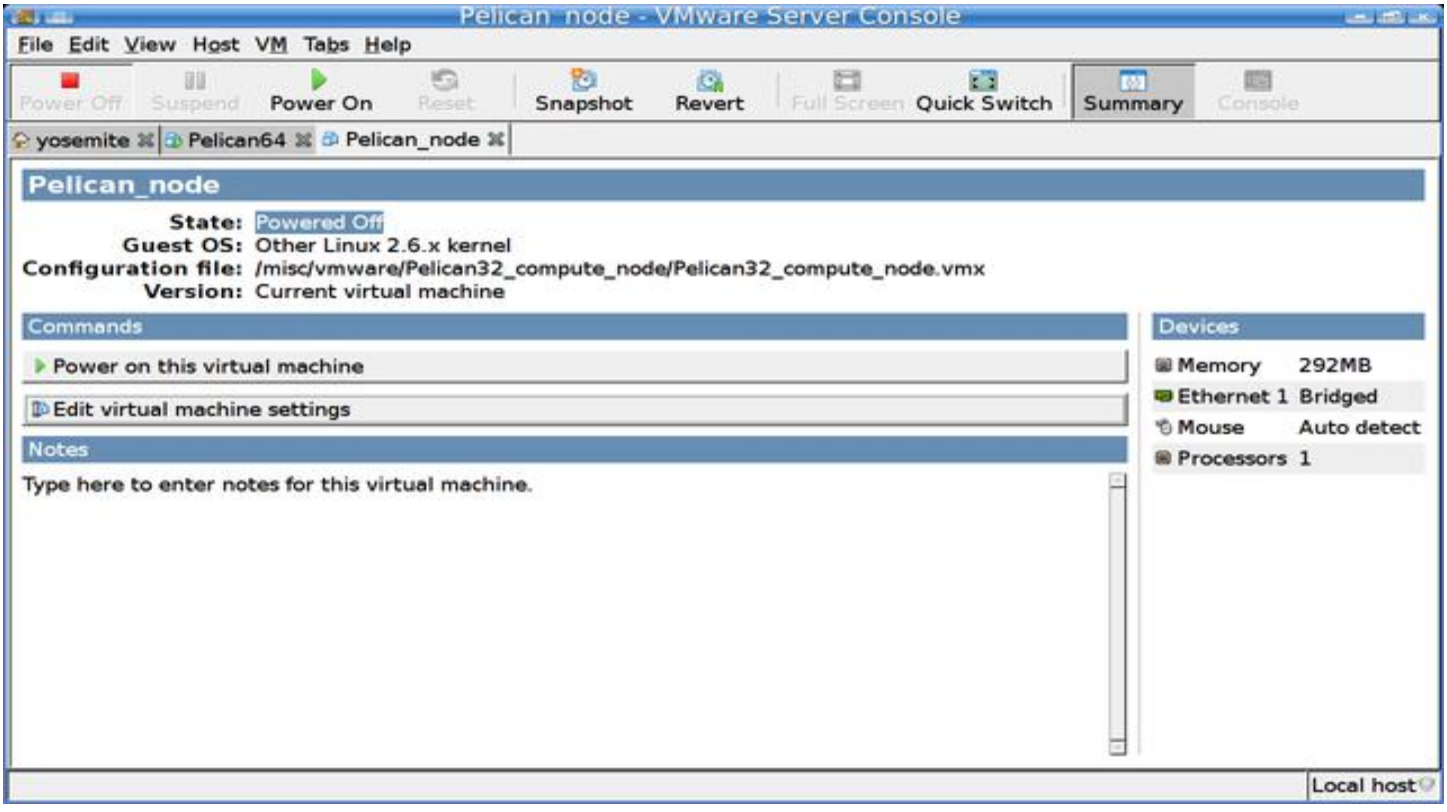


As is says in the last shot, you need to type "pelican_setup" to start dhcp, nfs, etc., so that the compute nodes may be booted. If you type this, you will see the following, supposing that you have more than 1 network device. (Note, you can enter the graphical environment by typing "startx" and then set the cluster up from there, if you prefer).

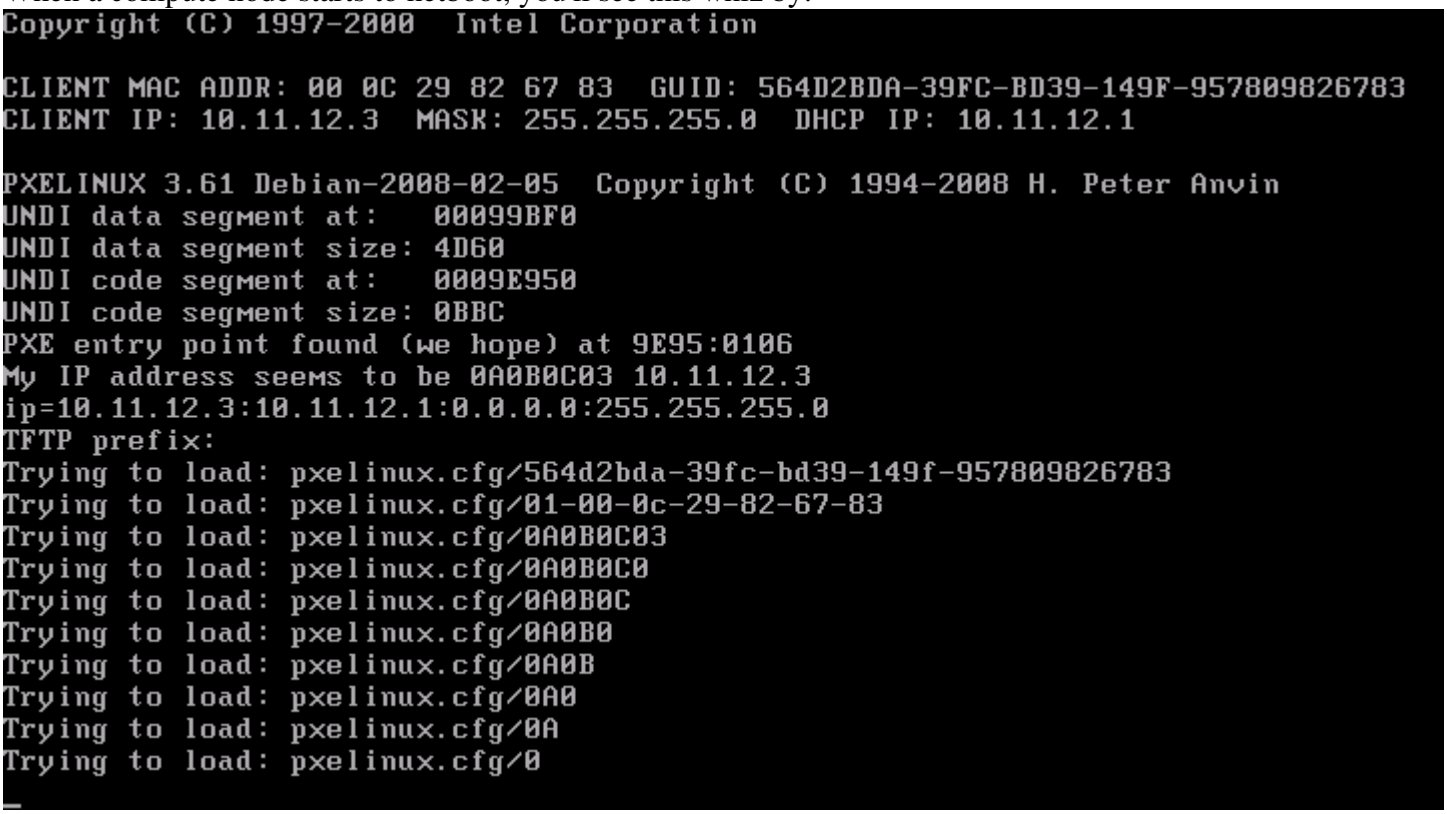
After you choose the net device, services are started. When you see the following screen, you can go turn on

the compute nodes. Choose "yes".

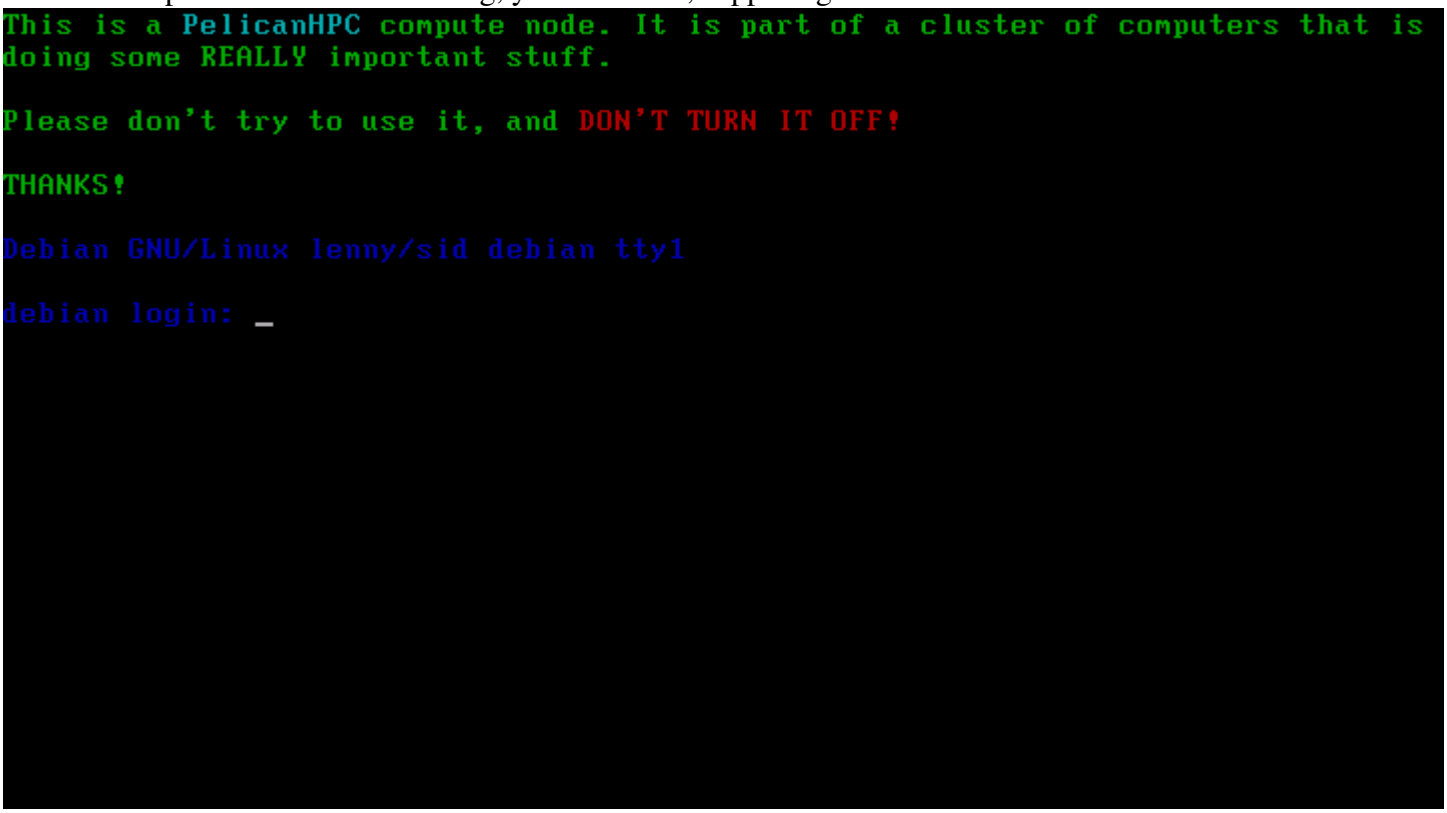
Here's a shot of a virtual cluster setup, the frontend node is running in one tab, and the compute node is ready to be turned on in the other tab. If the frontend node is virtual and you would like to boot real compute nodes, be sure to specify that the virtual network device that connect to the cluster is "bridged". To have internet access on the virtual frontend node, add a second network device with "NAT" networking, to share your real Internet connection.



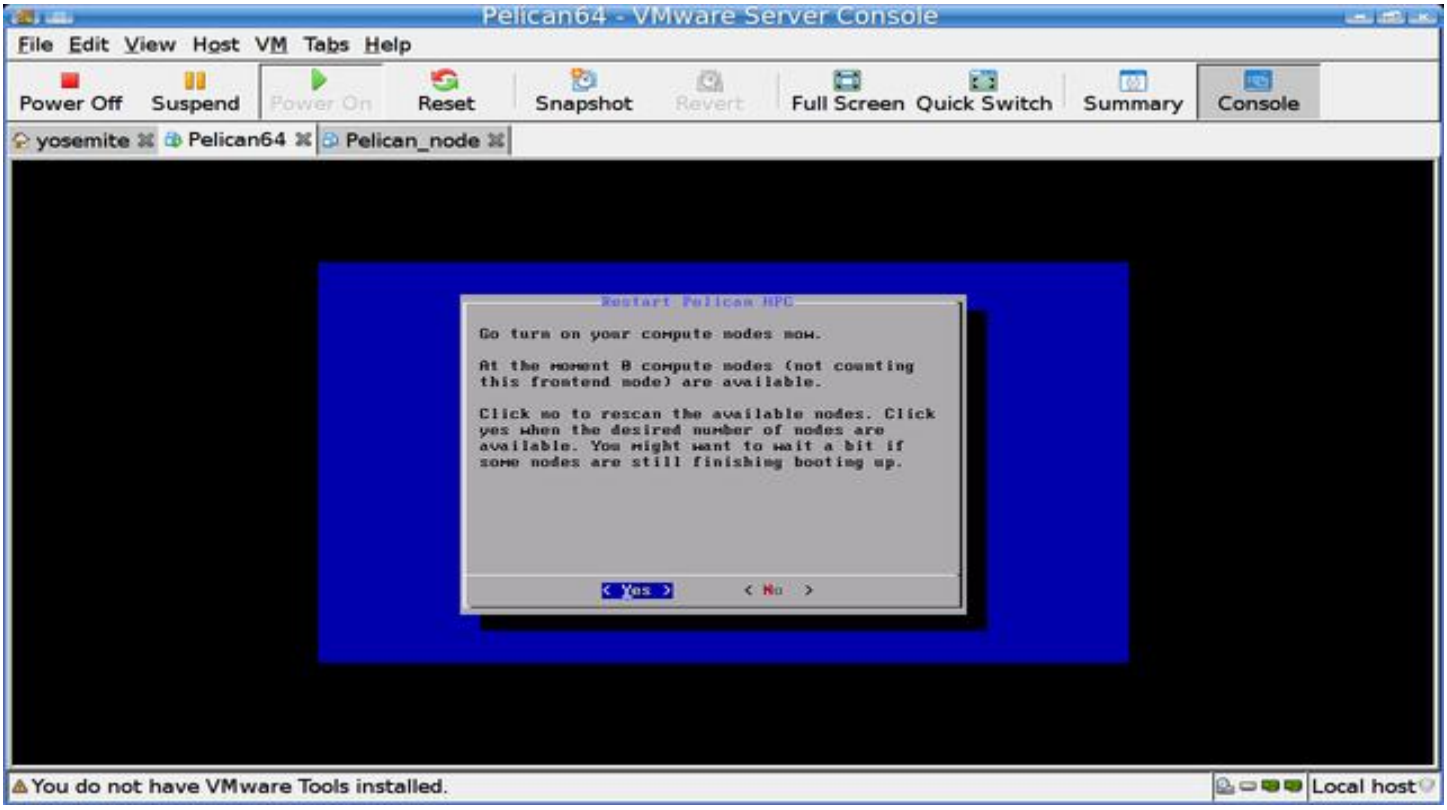
When a compute node starts to netboot, you'll see this whiz by:



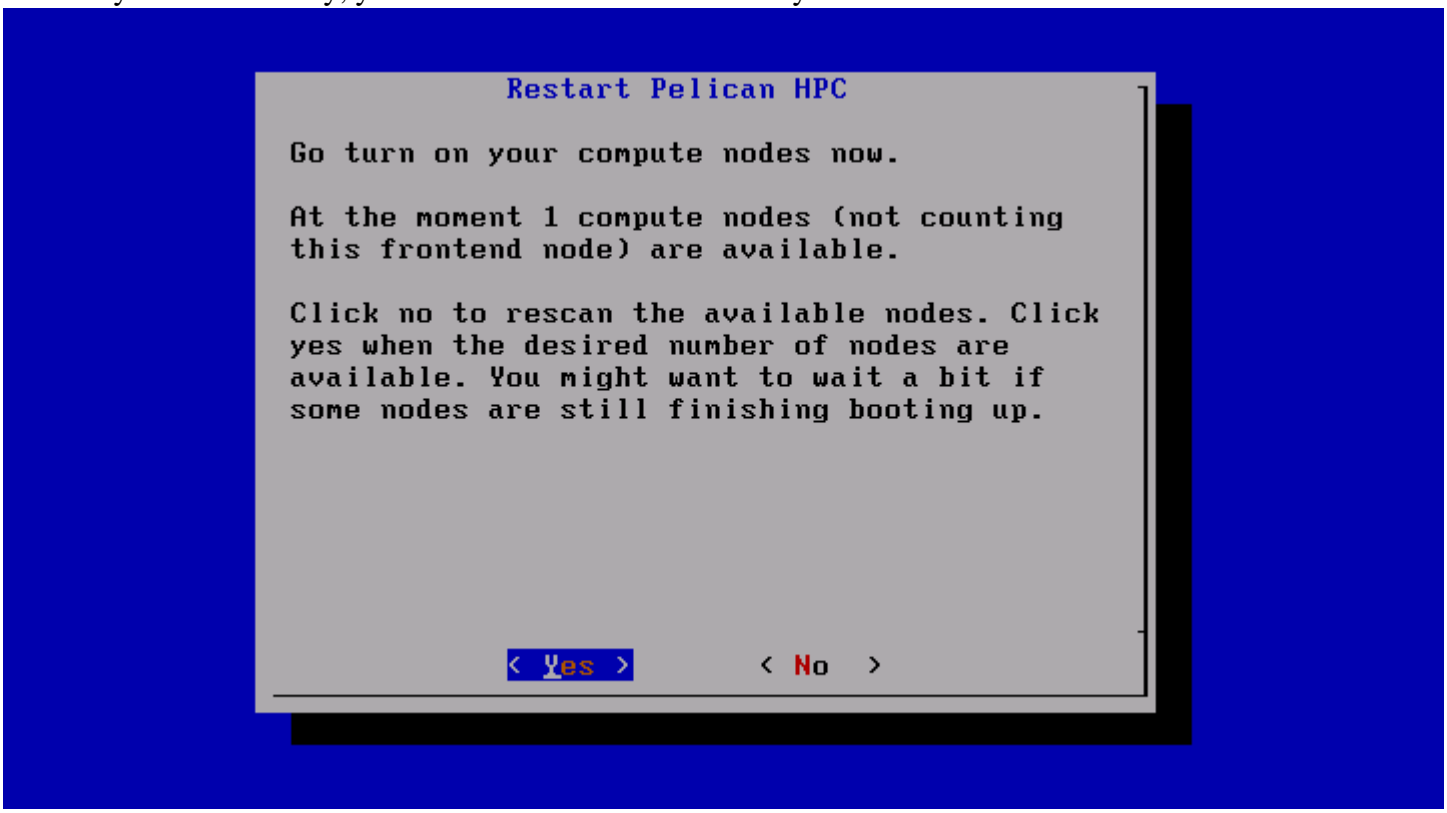
When a compute node is done booting, you'll see this, supposing that it has a monitor:



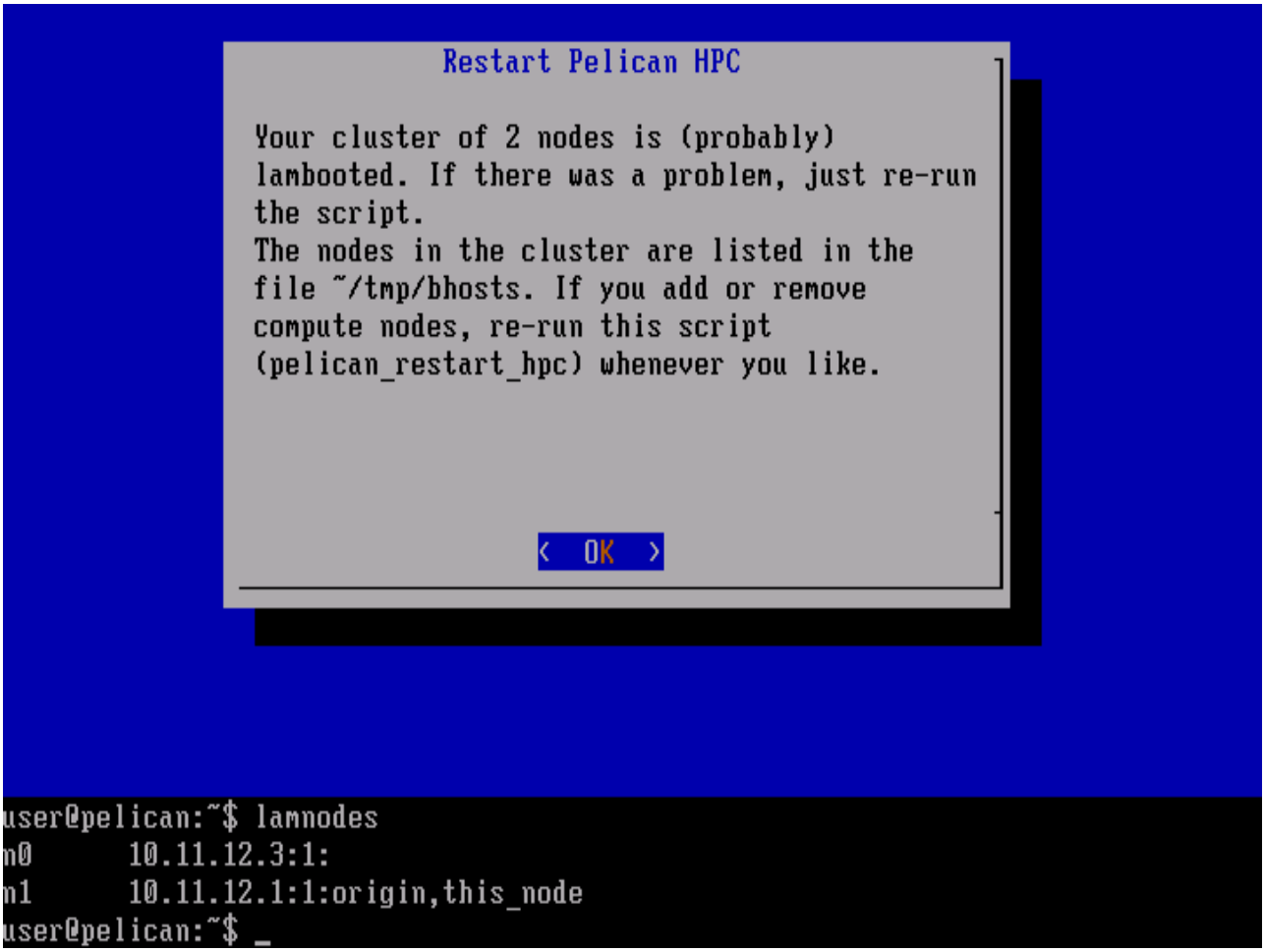
Here's a shot of the virtual cluster, with the frontend node and a compute node. The compute node has not yet reported itself to be available (count is zero).



Here's a larger shot of the same thing you see in the last shot. Now the count is 1, which means that the compute node has booted. Keep choosing "no" until all of your compute nodes are accounted for. Then choose "yes". Don't worry, you can add nodes in the future if you like.



Once you click yes, you'll see something like the following, depending on how many nodes you have. Note how I type "lamnodes" to check that it really worked.



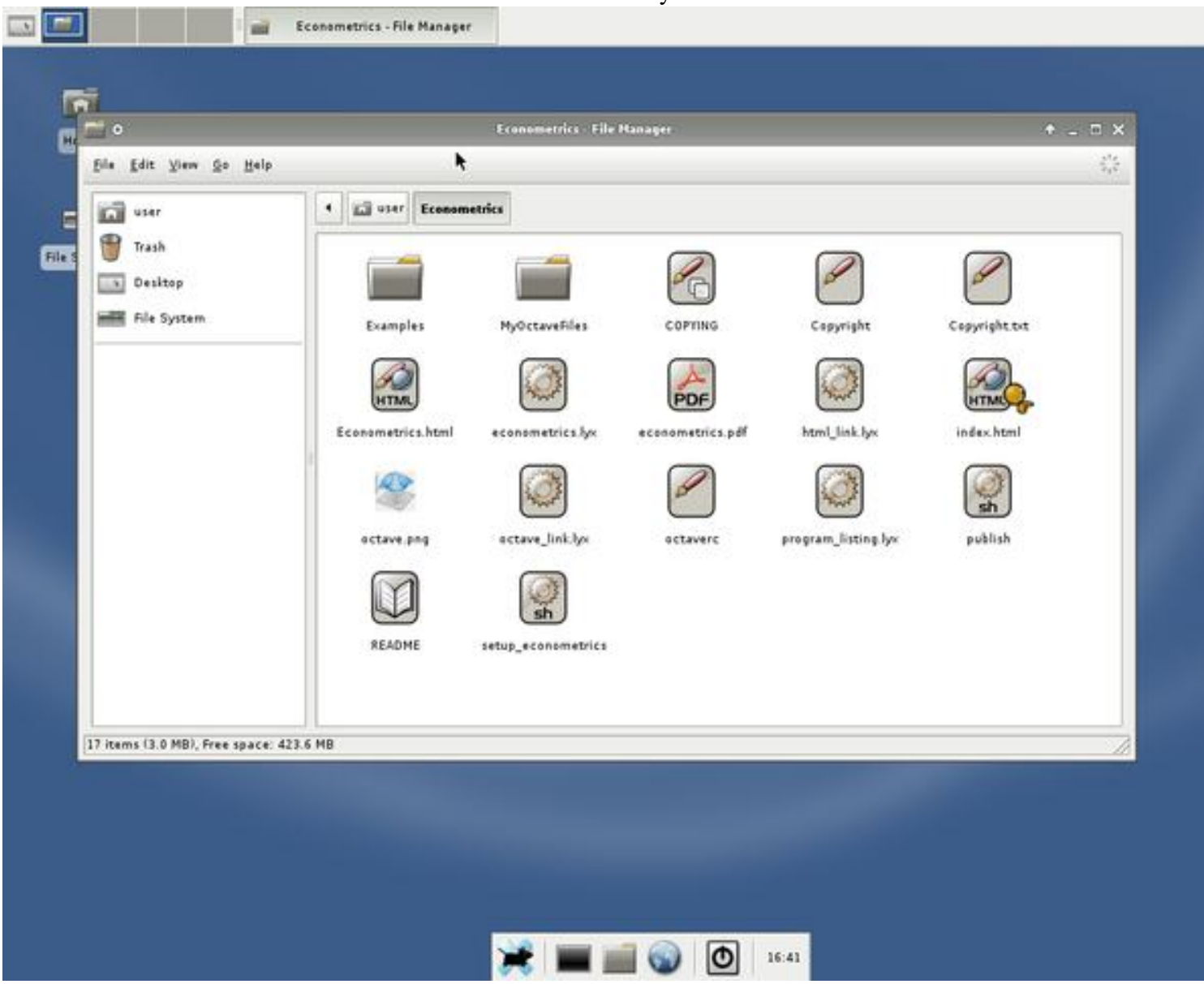
OK, that's it, the cluster is ready to use. Some other tips:

- change the password on all nodes. Really do this if your machines are connected to any external network. I revoke your license to use PelicanHPC if you're not smart enough to do this. The default password of the released ISO images is "live". Serious users will probably want to make their own version of PelicanHPC with a private password, using the make_pelican script.
- you can add software to the frontend node using "apt-get install whatever", supposing that the frontend has a second net card that you have configured to enable Internet access.
- the default MPI setup is in the file /home/user/tmp/bhosts. This assigns ranks to hosts in a round robin fashion. If your hosts have different speeds, numbers of cores, etc., you should modify this file and re-lamboot (if you use LAM).
- ksysguard is available, and a small amount of effort will turn it into a nice cluster monitor. See this post for general information on how to do it.
- if you need other packages, would like to set your own default password, use a storage device to mount as /home, etc., then you can make your own version pretty easily using the make_pelican script that is available on the PelicanHPC homepage. Hopefully, I'll write some documentation for that in the near future. But you can probably figure it out if you really need it :-)
- You can resize the cluster (add or remove compute nodes) whenever you like, by running "pelican_restarthpc".

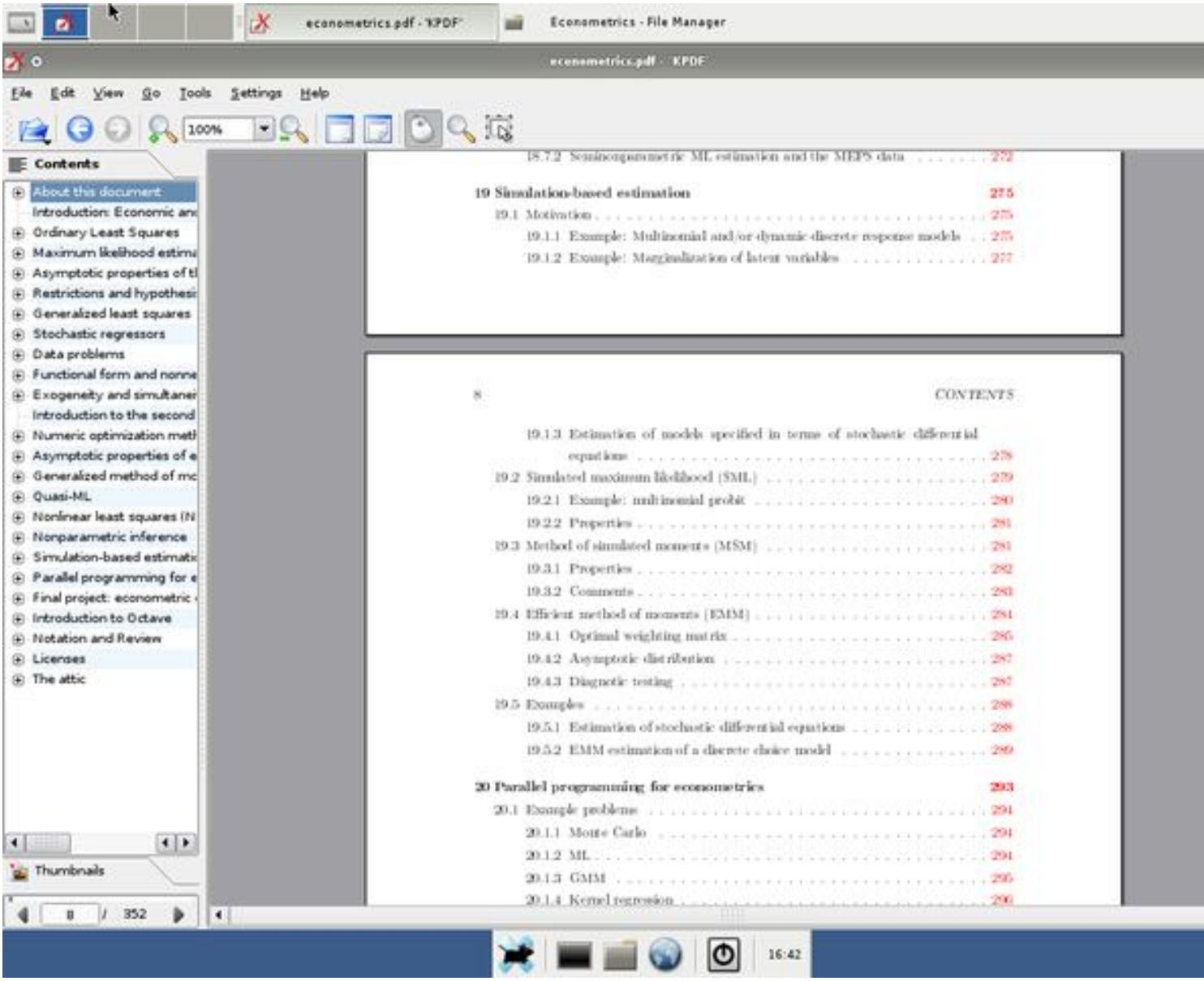
Return to contents

Example software

PelicanHPC has the Linpack HPL benchmark, and some extensive examples from the field of econometrics that use MPITB for GNU Octave. Econometrics is a field of study that applies statistical methods to economic models. The software is in the Econometrics directory:

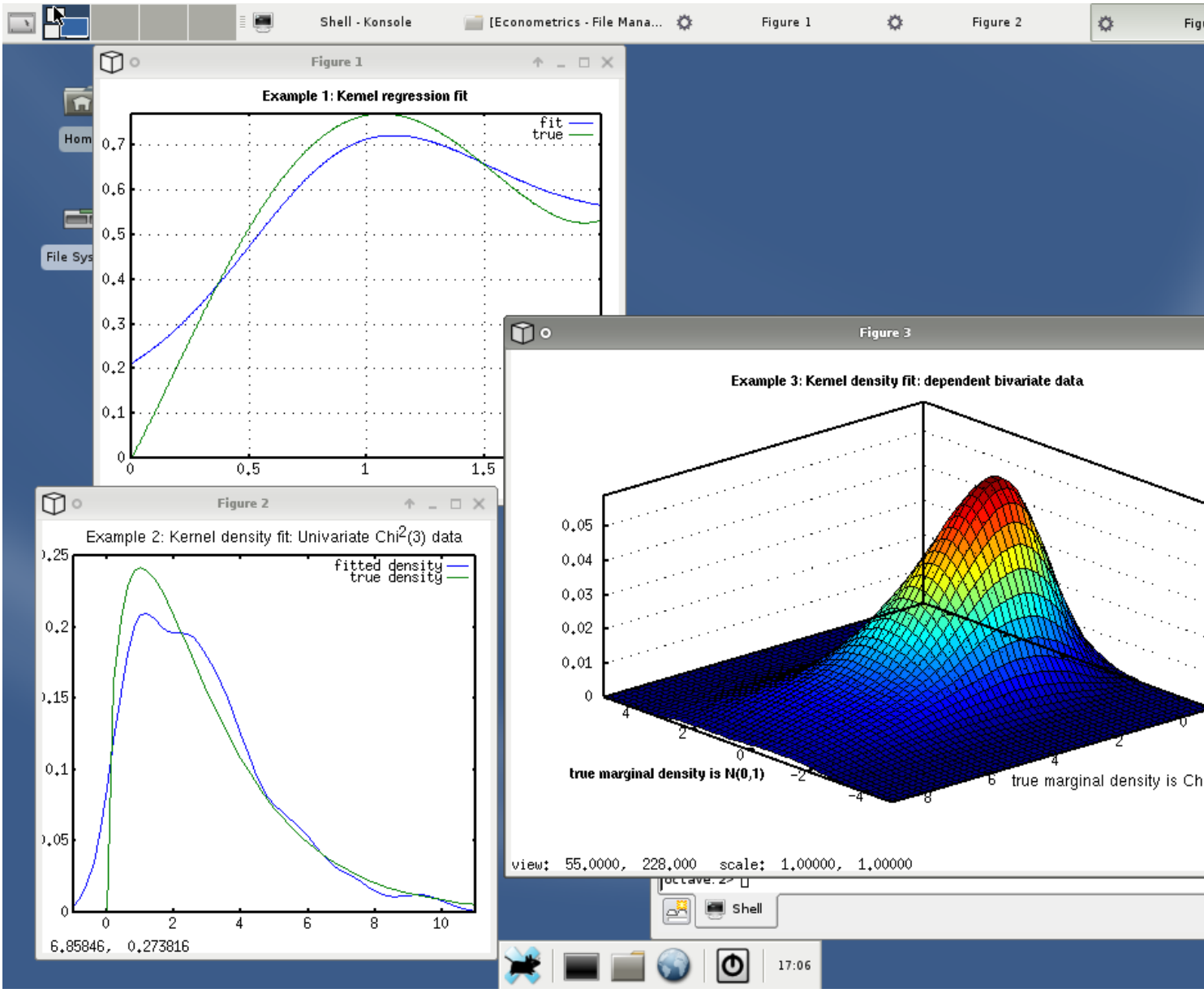


There is a document "econometrics.pdf" that has a lot of information, including some about parallel computing:



Open a terminal, type "octave" and then "kernel_example" (please note that underscore back there):

et viola! some nice pictures:

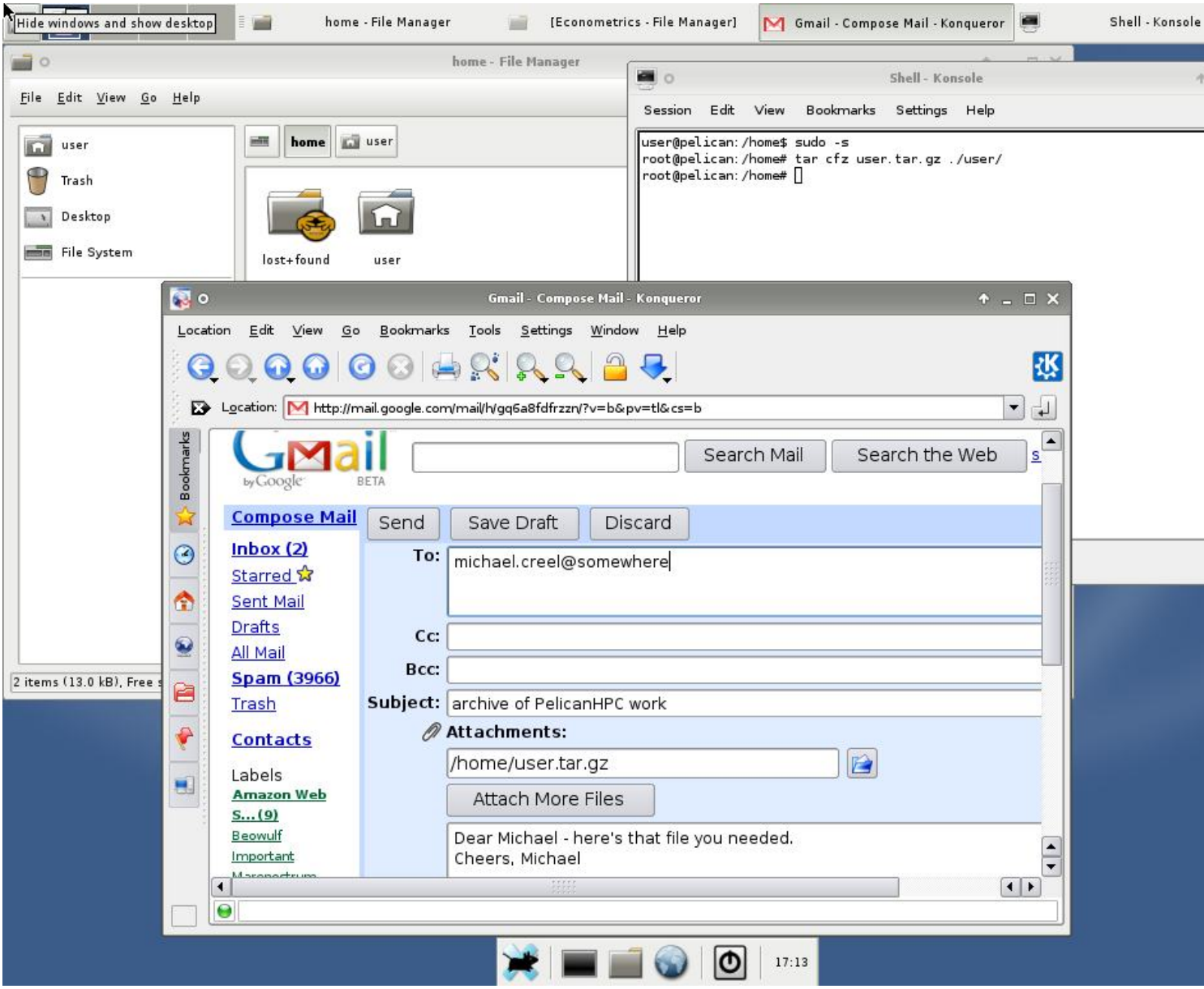


Other things to try are "pea_example", "bfgsmin_example", "mle_example", "gmm_example", "mc_example1", "mc_example2" and a few others I'm forgetting about. To find where the code is, type "help mc_example1", for example, while in Octave. Then go edit the relevant file to learn more about what it does.

Return to contents

Saving your work

By default, PelicanHPC images put /home/user on a ramdisk which disappears when you shut down. You need to save your work between sessions, if you want to re-use it. There are many options, such as mounting a hard disk, using a USB device, etc. If you have an Internet connection configured, you can email it to yourself, as is illustrated in the next shot:



If you use PelicanHPC for serious work, I highly recommend mounting a storage device to use as /home, so that your work will be saved between sessions without taking any special steps. An example of the commands you could use to do this would be as follows.

```
sudo -s
mkdir /junk
mount /dev/YOURDEV /junk
cp -a /home/user /junk
mount --bind /junk /home
exit
exit
```

The variations and possibilities here are so numerous that I don't want to attempt to explain this further. Don't try to do this until you know what you're doing.

Return to contents

Using the make_pelican script

The distributed ISO images provide a bare bones cluster setup system, plus some packages that I use in my research and teaching. There are a few examples taken from my work, which may be of interest to those learning the basics of MPI, or to people interested in econometrics. However, many users will find that Pelican does not contain packages that they need, so a means of customizing the CD image is required. PelicanHPC is made by running a single script "make_pelican", which is available on the download page. If you have the prerequisites for running the script, it is very easy to make a customized version of Pelican. The prerequisites are:

- a Debian or Debian-like (Ubuntu, etc.) installation of GNU/Linux. This can be a minimal installation in a chroot jail if you prefer to run something else for your normal work. You could even use a virtual machine under Windows, if you are a Windows user.
- the live-helper package should be obtained from this source and installed on your Debian or Debian-like system using "dpkg -i live-helper<tab>"
- examine the make_pelican script, which contains some self-explanatory comments. Add the packages you need to the package list section. Note that you can choose a different password here.
- you need to run the make_pelican script as the root user (maybe not, but I haven't checked). A fast internet connection is helpful, since a lot of packages need to be downloaded. Also, it helps to build the image on a fast, hopefully multicore computer. Parts of the build process are parallelized and will take advantage of multiple cores. Build time for the default configuration on a decent dual core laptop with lot of RAM is about half an hour.
- when you are done, there will be a file "binary.iso" in the ../<architecture>/frontend directory, where ../ is the location of the make_pelican script, and <architecture> is either i386 or amd64, depending on which you left uncommented in the script.
- There is a manual for Debian Live. Please have a look at it before trying to use make_pelican. Additional information is at http://debian-live.alioth.debian.org/. This information is the main documentation, since make_pelican is just a script that provides a specific configuration to the Debian Live system of building a live CD image. Also remember that "man live-helper", "man lh_config" and "man lh_build" will give you information.